Contents

0-8493-0052-5/00/$0.00+$.50

Chapter 1

Performance Characteristics of Potential Petascale Scientific Applications

Leonid Oliker, John Shalf, Jonathan Carter, Andrew Canning, Shoaib Kamil, Michael Lijewski

CRD/NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Stéphane Ethier

Princeton Plasma Physics Laboratory, Princeton University, Princeton, NJ 08453

Abstract After a decade where HEC (high-end computing) capability was dominated by the rapid pace of improvements to CPU clock frequency, the performance of next-generation supercomputers is increasingly differentiated by varying interconnect designs and levels of integration. Understanding the tradeoffs of these system designs, in the context of high-end numerical simulations, is a key step towards making effective petascale computing a reality. This work represents one of the most comprehensive performance evaluation studies to date on modern HEC systems, including the IBM Power5, AMD Opteron, IBM BG/L, and Cray X1E. A novel aspect of our study is the emphasis on full applications, with real input data at the scale desired by computational scientists in their unique domain. We examine five candidate ultra-scale applications, representing a broad range of algorithms and computational structures. Our work includes the highest concurrency experiments to date on five of our six applications, including 32K processor scalability for two of our codes and describe several successful optimizations strategies on BG/L, as well as improved X1E vectorization. Overall results indicate that our evaluated codes have the potential to effectively utilize petascale resources; however, several applications will require reengineering to incorporate the additional levels of parallelism necessary to utilize the vast concurrency of upcoming ultra-scale systems.

0-8493-0052-5/00/$0.00+$.50

1.1 Introduction

Computational science is at the dawn of petascale computing capability, with the potential to achieve simulation scale and numerical fidelity at hitherto unattainable levels. However, harnessing such extreme computing power will require an unprecedented degree of parallelism both within the scientific applications and at all levels of the underlying architectural platforms. Unlike a decade ago — when the trend of HEC (high-end computing) systems was clearly towards building clusters of commodity components — today one sees a much more diverse set of HEC models. Increasing concerns over power efficiency is likely to further accelerate recent trends towards architectural diversity through new interest in customization and tighter system integration. Power dissipation concerns are also driving HPC system architectures from the historical trend of geometrically increasing clock rates towards geometrically increasing core counts (multicore), leading to daunting levels of concurrency for future petascale systems. Employing an even larger number of simpler processor cores operating at a lower clock frequency, as demonstrated by BG/L, offers yet another approach to improving the power efficiency of future HEC platforms. Understanding the tradeoffs of these computing paradigms, in the context of high-end numerical simulations, is a key step towards making effective petascale computing a reality. The main contribution of this work is to quantify these tradeoffs by examining the effectiveness of various architectural models for HEC with respect to absolute performance and scalability across a broad range of key scientific domains.

A novel aspect of our effort is the emphasis on full applications, with real input data at the scale desired by computational scientists in their unique domain, which builds on our previous efforts [5, 18–20] and complements a number of other related studies [6, 11, 17, 31]. Our application suite includes a broad spectrum of numerical methods and data-structure representations in the areas of Magnetic Fusion (GTC), Astrophysics (CACTUS), Fluid Dynamics (ELBM3D), Materials Science (PARATEC), and AMR Gas Dynamics (HyperCLaw). We evaluate performance on a wide range of architectures with varying degrees of component customization, integration, and power consumption, including: the Cray X1E customized parallel vector-processor, which utilizes a tightly-coupled custom interconnect; the commodity IBM Power5 and AMD Opteron processors integrated with custom fat-tree based Federation and 3D-torus based XT3 interconnects, respectively; the commodity Opteron processor integrated with the InfiniBand high-performance commodity network; and the IBM Blue Gene/L (BG/L) which utilizes a customized SOC (system on chip) based on commodity, low-power embedded cores, combined with multiple network interconnects.

This work represents one of the most comprehensive performance evaluation studies to date on modern HEC platforms. We present the highest concurrency results ever conducted for our application suite, and show that the BG/L can attain impressive scalability characteristics all the way up to 32K processors on two of our applications. We also examine several application optimizations, including BG/L processor and interconnect-mappings for the SciDAC [23] GTC code, which achieve significant performance improvements over the original superscalar version. Additionally, we implement several optimizations for the HyperCLaw AMR calculation, and show significantly improved performance and scalability on the X1E vector platform, compared with previously published studies. Overall, we believe that these comprehensive evaluation efforts lead to more efficient use of community resources in both current installations and in future designs.

1.2 Target Architectures

Our evaluation testbed uses five different production HEC systems. Table 1.1 presents several key performance parameters of *Bassi*, *Jacquard*, *BG/L*, *Jaguar*, and *Phoenix*, including: STREAM benchmark results [27] showing the measured EP-STREAM [13] triad bandwidth when all processors within a node simultaneously compete for main memory; the ratio of STREAM bandwidth to the peak computational rate; the measured inter-node MPI latency [7]; and the measured bidirectional MPI bandwidth per processor pair when each processor simultaneously exchanges data with a distinct processor in another node.

Name	Arch	Network	Network Topology	P/ Node	Clock (GHz)	Peak (GF/s/P)	Stream BW (GB/s/P)	Stream (B/F)[‡]	MPI Lat (μsec)	MPI BW (GB/s/P)
Jaguar	Opteron	XT3	3D Torus	1	2.6	5.2	2.5	0.48	5.5[*]	1.2
Jacquard	Opteron	InfiniBand	Fat-tree	2	2.2	4.4	2.3	0.51	5.2	0.73
Bassi	Power5	Federation	Fat-tree	8	1.9	7.6	6.8	0.85	4.7	0.69
BG/L	PPC440	Custom	3D Torus	2	0.7	2.8	0.9	0.31	2.2[†]	0.16
Phoenix	X1E	Custom	4D-Hcube	4	1.1	18.0	9.7	0.54	5.0	2.9

TABLE 1.1: Architectural highlights of studied HEC platforms. An MSP is defined as a processor for the X1E data.

Bassi: Federation/Power5: The Power5-based Bassi system is located at Lawrence Berkeley National Laboratory (LBNL) and contains 122 8-way SMP nodes interconnected via a two-link network adapter to the IBM Federation HPS switch. The 1.9 GHz RISC Power5 processor contains a 64KB instruction cache, a 1.9MB on-chip L2 cache as well as a 36MB on-chip L3 cache. The IBM custom Power5 chip has two Synchronous Memory Interface (SMI) chips, aggregating four DDR 233 MHz channels for an impressive measured STREAM performance of 6.8GB/s per processor. The Power5 includes an integrated memory controller and integrates the distributed switch fabric between the memory controller and the core/caches. The experiments in this work were conducted under AIX 5.2. Several experiments were also conducted on the 1532 node (12,256 processor) Power5-based Purple system at Lawrence Livermore National Laboratory.

Jacquard: InfiniBand/Opteron: The Opteron-based Jacquard system is also located at LBNL. Jacquard contains 320 dual-processor nodes and runs Linux 2.6.5 (PathScale 2.0 compiler). Each node contains two 2.2 GHz Opteron processors, interconnected via InfiniBand fabric in a fat-tree configuration. The Opteron uses a SIMD floating-point unit accessed via the SSE2 instruction set extensions, and can execute two double-precision floating-point operations per cycle. The processor possesses an on-chip DDR memory controller as well as tightly-integrated switch interconnection

[‡]Ratio of measured Stream bandwidth to peak processor computational rate.
[*]Minimum latency for the XT3 torus. There is a nominal additional latency of $50ns$ per hop through the torus.
[†]Minimum latency for the BG/L torus. There is an additional latency of up to $69ns$ per hop through the torus.

via HyperTransport. Jacquard's Infiniband network utilizes 4X single-data-rate links, configured in a 2-layer CLOS/Fat-tree configuration using Mellanox switches.

Jaguar: XT3/Opteron: The Jaguar experiments were conducted on the 5,202 node XT3 system operated by the National Leadership Computing Facility (NLCF) at Oak Ridge National Laboratory running the Catamount microkernel on the compute nodes. Each node of the Cray XT3 contains a single, dual-core 2.6 GHz AMD Opteron processor. The processors are tightly-integrated to the XT3 interconnect via a Cray SeaStar ASIC through a 6.4GB/s bidirectional HyperTransport interface. All the SeaStar routing chips are interconnected in a 3D torus topology, where — similarly to the BG/L system — each node has a direct link to six of its nearest neighbors on the torus with a peak bidirectional bandwidth of 7.6GB/s. Note that Jaguar is similar to Jacquard in terms of the processor architecture and peak memory bandwidth; however, in addition to having vastly different interconnect technologies, Jaguar uses dual-core technology as opposed to Jacquard's single-core processors.

BG/L: Multiple Networks/PPC440: The IBM Blue Gene/L (BG/L) system represents a unique approach to supercomputing design, which allows unprecedented levels of parallelism, with low unit cost and power consumption characteristics. Our work presents performance results on the 1024-node BG/L located at Argonne National Laboratory (ANL) running OS SLES9, as well as the 20K-node system at IBM's Watson Research Center system (BGW), currently the world's third most powerful supercomputer [29] (16K nodes available at time of testing). Each BG/L node contains two 700MHz PowerPC 440 processors, on-chip memory controller and communication logic, and only 512MB of memory. The CPU's dual FPUs (called double hummer) are capable of dispatching two MADDs per cycle, for a peak processor performance of 2.8 Gflops/s. However, the second FPU is not an independent unit, and can only be used with special SIMD instructions — thus, making it difficult for the core to perform at close to peak except for specially hand-tuned code portions. Our experiments primarily examine performance in *coprocessor mode* where one core is used for computation and the second is dedicated to communication. Additionally, several experiments were conducted using 32K processors of BGW in *virtual node mode* where both cores are used for both computation and communication.

The BG/L nodes are connected via five different networks, including a torus, collective tree, and global interrupt tree. The 3D Torus interconnect is used for general-purpose point-to-point message passing operations using 6 independent point-to-point serial links to 6 nearest neighbors that operate at 175MB/s per direction (bidirectional) for an aggregate bandwidth of 2.1GB/s per node. The global tree collective network is designed for high-bandwidth broadcast operations (one-to-all) using three links that operate at a peak bandwidth of 350MB/s per direction for an aggregate 2.1GB/s bidirectional bandwidth per node. Finally, the global interrupt network provides fast barriers and interrupts with a system wide constant latency of $\approx 1.5\mu s$.

Phoenix: Custom Network/X1E: The Cray X1E is the recently-released follow-on to the X1 vector platform. Vector processors expedite uniform operations on independent data sets by exploiting regularities in the computational structure. The X1E computational core, called the single-streaming processor (SSP), contains two 32-stage vector pipes running at 1.13 GHz. Each SSP contains 32 vector registers holding 64 double-precision words, and operates at 4.5 Gflop/s. The SSP also contains a two-way out-of-order superscalar processor (564 MHz) with two 16KB caches (instruction and data). Four SSPs can be combined into a logical computational unit called the multi-streaming processor (MSP), and share a 2-way set associative 2MB data Ecache, with a peak performance of 18Gflops/s. Note that the scalar unit operates at 1/4th the peak of SSP vector performance, but offers effectively 1/16 MSP performance if a loop can neither be multistreamed nor vectorized. The X1E interconnect is hierarchical, with subsets of 16 SMP nodes (each containing 8 MSPs) connected via a crossbar, these subsets are connected in a 4D-hypercube topology. All reported X1E experiments were performed on a 768-MSP system running UNICOS/mp 3.0.23 and operated by Oak Ridge National Laboratory.

1.3 Scientific Application Overview

Name	Lines	Discipline	Methods	Structure
GTC	5,000	Magnetic Fusion	Particle in Cell, Vlasov-Poisson	Particle/Grid
CACTUS	84,000	Astrophysics	Einstein Theory of GR, ADM-BSSN	Grid
ELBD	3,000	Fluid Dynamics	Lattice Boltzmann, Navier-Stokes	Grid/Lattice
PARATEC	50,000	Materials Science	Density Functional Theory, FFT	Fourier/Grid
HyperCLaw	69,000	Gas Dynamics	Hyperbolic, High-order Godunov	Grid AMR

TABLE 1.2: Overview of scientific applications examined in our study.

Five applications from diverse areas in scientific computing were chosen to compare the performance of our suite of leading supercomputing platforms. We examine: GTC, a magnetic fusion application that uses the particle-in-cell approach to solve non-linear gyrophase-averaged Vlasov-Poisson equations; Cactus, an astrophysics framework for high-performance computing that evolves Einstein's equations from the Theory of General Relativity; ELBM3D, a lattice-Boltzmann code to study turbulent fluid flow; PARATEC, a first principles materials science code that solves the Kohn-Sham equations of density functional theory to obtain electronic wave functions; HyperCLaw, an adaptive mesh refinement (AMR) framework for solving the Hyperbolic conservation laws of gas dynamics via a higher-order Godunov method. Table 1.2 presents an overview of the application characteristics from our evaluated simulations.

These codes are candidate ultra-scale applications with the potential to fully utilize leadership-class computing systems, and represent a broad range of algorithms and computational structures. Communication characteristics include: nearest-neighbor and allreduce communication across the toroidal grid and poloidal grid (respectively) for the particle-in-cell GTC calculation; simple ghost boundary exchanges for the stencil-based ELBM3D and Cactus computations; all-to-all data transpositions used to implement PARATEC's 3D FFTs, and complex data movements required to create and dynamically adapt grid hierarchies in HyperCLaw. Examining these varied computational methodologies across a set of modern supercomputing platforms allows us to study the performance tradeoffs of different architectural balances and topological interconnect approaches.

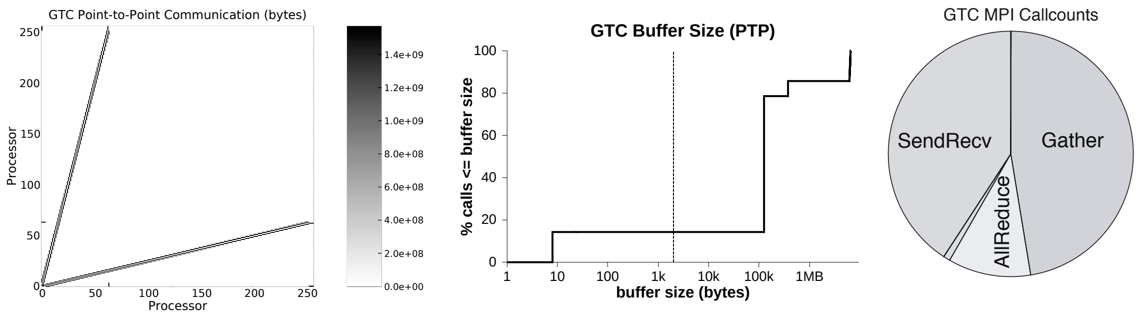
To study the topological connectivity of communication for each application, we utilize the IPM [26] tool, recently developed at LBNL. IPM is an application profiling layer that allows us to non-invasively gather the communication characteristics of these codes as they are run in a production environment. By recording statistics on these message exchanges we can form an undirected graph which describes the topological connectivity required by the application. We use the IPM data to create representations of the topological connectivity of communication for each code as a matrix — where each point in the graph indicates message exchange and (color coded) intensity between two given processors — highlighting the vast range of communication requirements within our application suite. IPM is also used to collect statistics on MPI utilization and the sizes of point-to-point messages, allowing us to quantify the fraction of messages bandwidth- or latency-bound. We dividing line between these two regimes is an aggressively designed bandwidth-delay product of 2Kb. See [14, 24] for a more detailed explanation of the application communication characteristics, data collection methodology, and bandwidth-delay product thresholding.

Experimental results show either strong scaling (where the problem size remains fixed regardless of concurrency), or weak scaling (where the problem size grows with concurrency such that the per-processor computational requirement remains fixed) — whichever is appropriate for a given application's large-scale simulation. Note these applications have been designed and highly optimized on superscalar platforms; thus, we describe newly devised optimizations for the vector platforms where appropriate. Performance results measured on these systems, presented in Gflop/s per processor (denoted as Gflops/P) and percentage of peak, are used to compare the time to solution of our evaluated platforms. The Gflop/s value is computed by dividing a valid baseline flop-count by the measured wall-clock time of each platform — thus the ratio between the computational rates is the same as the ratio of runtimes across the evaluated systems. All results are shown using the fastest (optimized) available code versions.

1.4 GTC: Particle-in-Cell Magnetic Fusion

GTC is a 3D particle-in-cell code developed for studying turbulent transport in magnetic confinement fusion plasmas [8, 15]. The simulation geometry is that of a torus, which is the natural configuration of all tokamak fusion devices. As the charged particles forming the plasma move within the externally-imposed magnetic field, they collectively create their own self-consistent electrostatic (and electromagnetic) field that quickly becomes turbulent under driving temperature and density gradients. The particle-in-cell (PIC) method describes this complex interaction between fields and particles by solving the 5D gyro-averaged kinetic equation coupled to the Poisson equation. In the PIC method, the interaction between particles is calculated using a grid on which the charge of each particle is deposited and then used in the Poisson equation to evaluate the field. This is the scatter phase of the PIC algorithm. Next, the force on each particle is gathered from the grid-base field and evaluated at the particle location for use in the time advance.

An important approximation in GTC comes from the fact that fast particle motion along the magnetic field lines leads to a quasi-two-dimensional structure in the electrostatic potential. Thus, the Poisson equation needs only to be solved on a 2D poloidal plane. GTC utilizes a simple iterative solver since the linear system is diagonally dominant [16]; note that this differs from most PIC techniques that solve a spectral system via FFTs. For 10 particles per cell per processor, the solve phase accounts only for 6% of the compute time, a very small fraction compared to the 85% spent in the scatter and gather-push phases. The primary direction of domain decomposition for GTC is in the toroidal direction in order to take advantage of the locality of the 2D Poisson solve. GTC uses a logically non- rectangular field-aligned grid, in part, to keep the number of particles



(b) (c)

FIGURE 1.1: GTC (a) communication topology and intensity for point-to-point messaging (b) cumulative distribution function of point-to-point message buffer sizes with horizontal dividing line representing the border between bandwidth bound (right of line) and latency bound (left of line) message sizes and (c) breakdown of MPI calls counts.

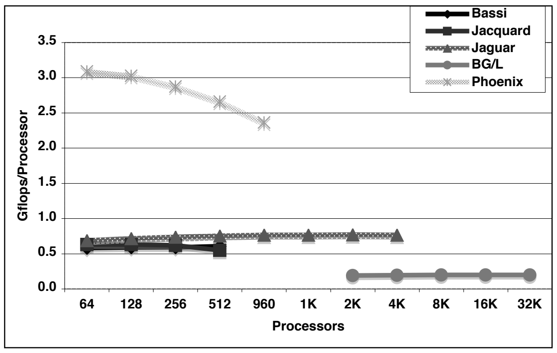
per cell nearly constant. The mesh effectively twists along with the magnetic field in the toroidal direction.

GTC utilizes two levels of distributed memory parallelism. The first — based on the original GTC implementation — is a one-dimensional domain decomposition in the toroidal direction (long way around the torus) while the second is a particle distribution within each toroidal slice. The processors in charge of the same toroidal slice have a complete copy of the local grid. However, each processor within the toroidal slice holds only a fraction of the particles, and data is exchanged between slices via a local MPI sub-communicator when updating grid quantities. In the toroidal direction, a separate MPI sub-communicator supports communication between the toroidal slices to support movement of particles between the slices.

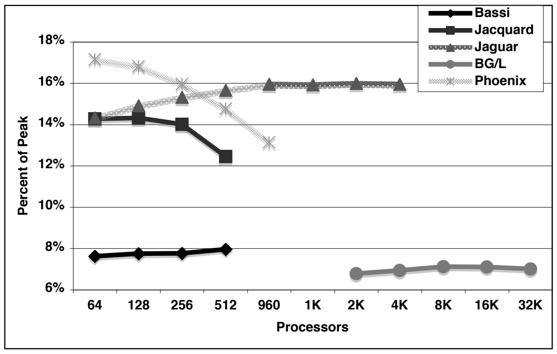
Figure 1.1 (a) shows the regular communication structure exhibited by GTC. This particle-in-cell calculation uses a one-dimensional domain decomposition across the toroidal computational grid, causing each processor to exchange data with its two neighbors as particles cross the left and right boundaries. Additionally, there is a particle decomposition within each toroidal slice as described above. Therefore, on average each MPI task communicates with 4 neighbors, so much simpler interconnect networks will be adequate for its requirements. The cumulative distribution function of message buffer sizes in Figure 1.1 (b), shows that the communication requirements are strongly bandwidth bound, where more than 80% of the message sizes are in the bandwidth bound regime — the horizontal reference line represents the border between bandwidth bound (right of line) and latency bound (left of line) message sizes, (see [24] for more details). The distribution of MPI call counts shown in Figure 1.1 (c) are strongly biased towards collective calls that have very small (<128 byte) latency bound messages sizes; at the petaflop scale level, scalable and efficient collective messaging will become an increasingly important factor for maintaining scalable performance for GTC and other PIC codes.

1.4.1 Experimental results

The benchmarks in our study were performed using 0.2 to 2 billion particles on a 2 million cell mesh (100 particles per cell per processor core). Figure 1.2 shows the results of a weak-scaling



(a)



(b)

FIGURE 1.2: GTC weak-scaling performance using 100 particles per cell per processor (10 for BG/L) in (a) runtime and (b) percentage of peak. All performance data on the BG/L system was collected in virtual node mode.

study of GTC on the platforms under comparison in both (a) raw performance and (b) percentage of peak. The size of the grid remains fixed since it is prescribed by the size of the fusion device being simulated, while the number of particles is increased in a way that keeps the same amount of work per processor for all cases.

Looking at the raw performance we see that the Phoenix platform clearly stands out with a Gflops/P rate up to 4.5 times higher than the second highest performer, the XT3 Jaguar. This was expected since the version of GTC used on Phoenix has been extensively optimized to take advantage of the multi-streaming vector processor [20]. In the latest improvements, the dimensions of the main arrays in the code have been reversed in order to speed up access to the memory banks, lead to higher performance. This change is not implemented in the superscalar version since it reduces cache reuse and hence slows down the code. Although still high, the performance per processor on the X1E decreases significantly as the number of processors, or MSPs, increases. This is probably due to the increase in intra-domain communications that arises when the number of processors per toroidal domain increases. An *Allreduce* operation is required within each domain to sum up the contribution of each processor, which can lead to lower performance in certain cases. Optimizing the processor mapping is one way of improving the communications but we have not explored this avenue on Phoenix yet.

Jacquard, Bassi, and Jaguar have very similar performance in terms of Gflops/P although Bassi is shown to deliver only about half the percentage of peak achieved on Jaguar, which displays outstanding efficiency and scaling all the way to 5184 processors. The percentage of peak achieved by particle-in-cell codes is generally low since the gather-scatter algorithm that characterizes this method involves a large number of random accesses to memory, making the code sensitive to memory access latency. However, the AMD Opteron processor used in both Jacquard and Jaguar delivers a significantly higher percentage of peak for GTC compared to all the other superscalar processors. It even rivals the percentage of peak achieved on the vector processor of the X1E Phoenix. This higher GTC efficiency on the Opteron is due, in part, to relatively low main memory latency access. On all systems other than Phoenix, GTC exhibits near perfect scaling, including up to 5K processors on Jaguar.

The percentage of peak achieved by GTC on BG/L is the lowest of the systems under study but the scalability is very impressive, all the way to 32,768 processors! The porting of GTC to the BG/L system was straightforward but initial performance was disappointing. Several optimizations were then applied to the code, most of them having to do with using BG/L-optimized math libraries such as MASS and MASSV to accelerate the performance of transcendental functions such as `sin()`, `cos()`, and `exp()`. Whereas the original code used the default GNU implementations for the transcendentals, the MASS and MASSV libraries include vector versions of those functions that can take advantage of improved instruction scheduling and temporal locality to achieve substantial improvements to the throughput of these operations in loops. [*] By replacing the default math intrinsics with their MASSV equivalents, we obtained a 30% increase in performance. Other optimizations consisted of loop unrolling and replacing calls to the Fortran `aint(x)` intrinsic function by `real(int(x))`. `aint(x)` results in a function call that is much slower than using the equivalent `real(int(x))`. These combined optimizations resulted in a performance improvement of almost 60% over original runs. We note that the results presented here are for virtual node mode where both cores on the node are used for computation. GTC maintains over 95% of its single-core (coprocessor-mode) performance when employing both cores on the BG/L node, which is quite promising as more cores will be added to upcoming processor roadmaps.

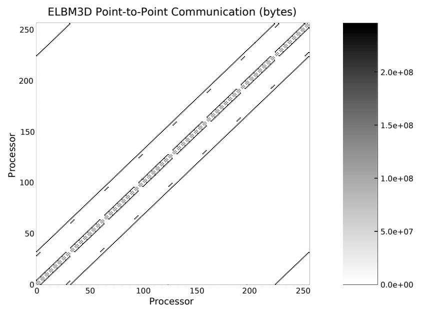
Another useful optimization performed on BGW was processor mapping. BG/L's 3D torus interconnect topology used for point-to-point communications is ideally suited for the toroidal geometry of GTC's computational domain. Additionally, the number of toroidal-slice domains used in the GTC simulations exactly match one of the dimensions of the BG/L network torus. Thus by using an explicit mapping file that aligns GTC's point-to-point communications along these toroidal slice domains (used for moving particles between the toroidal slices) to the BG/L link topology, the performance of the code improved by an additional 30% over the default communication mapping.

1.5 ELBM3D: Lattice Bolzmann Fluid Dynamics

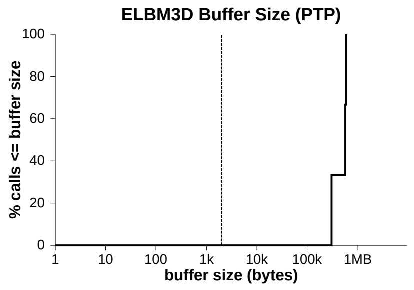
Lattice-Boltzmann methods (LBM) have proven a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids [28]. The basic idea is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties. These algorithms have been used extensively since the mid-1980s for simulating Navier-Stokes flows, and more recently extended to treat multi-phase flows, reacting flows, diffusion processes, and magneto-hydrodynamics. As can be expected from explicit algorithms, LBM are prone to numerical nonlinear instabilities as one pushes to higher Reynolds numbers. These numerical instabilities arise because there are no imposed constraints to enforce the distribution functions to remain non-negative. Entropic LBM algorithms, which do preserve the non-negativity of the distribution functions—even in the limit of arbitrary small transport coefficients—have recently been developed for Navier-Stokes turbulence [2], and are incorporated into a recently developed code [30].

While LBM methods lend themselves to easy implementation of difficult boundary geometries

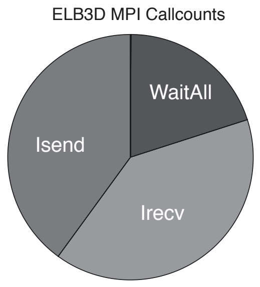
[*]The authors thank Bob Walkup for his BG/L optimization insights.



(a)



(b)



(c)

FIGURE 1.3: ELBM3D (a) communication topology and intensity (b) cumulative distribution function of point-to-point message buffer sizes and (c) breakdown of MPI calls counts.

(e.g., by the use of bounce-back to simulate no slip wall conditions), here we report on 3D simulations under periodic boundary conditions, with the spatial grid and phase space velocity lattice overlaying each other. Each lattice point is associated with a set of mesoscopic variables, whose values are stored in vectors proportional to the number of streaming directions. The lattice is partitioned onto a 3-dimensional Cartesian processor grid, and MPI is used for communication — a snapshot of the communication topology is shown in Figure 1.3(a), highlighting the relatively sparse communication pattern. As in most simulations of this nature, ghost cells are used to hold copies of the planes of data from neighboring processors.

For ELBM3D, a non-linear equation must be solved at each time step for each grid-point, so that the collision process satisfies certain constraints. Since this equation involves taking the logarithm of each component of the distribution function the whole algorithm becomes heavily constrained by the performance of the $log()$ function.

The connectivity of ELBM3D is shown in Figure 1.3(a) is structurally similar to Cactus, but exhibits a slightly different communication pattern due to the periodic boundary conditions of the code. This topology is not quite isomorphic to a mesh or toroidal interconnect topology, but would vastly underutilize the available bisection bandwidth of a fully-connected network like a fat-tree or crossbar. Figure 1.3(c) demonstrates that like Cactus, ELBM3D is dominated by point-to-point communication while Figure 1.3(b) shows that the point-to-point messages buffer sizes are very large and therefore strongly bandwidth bound.

1.5.1 Experimental results

We note that LBM codes are often compared on the basis of mega-updates-per-second as it is independent of the number of operations that the compiler generates for the LBM collision step. Our work, on the other hand, compares Gflop/s based on the FLOP count of a single baseline system and normalized on the basis of wallclock time (see Section 1.3). Thus the relative results maintain consistent comparisons of delivered performance between different system architectures offered by the mega-updates-per-second.

Strong-scaling results for a system of 512^3 grid points are shown in Figure 1.4 for both (a) raw performance and (b) percentage of peak. For each of the superscalar machines the code was restructured to take advantage of specialized `log()` functions — MASSV library for IBM and

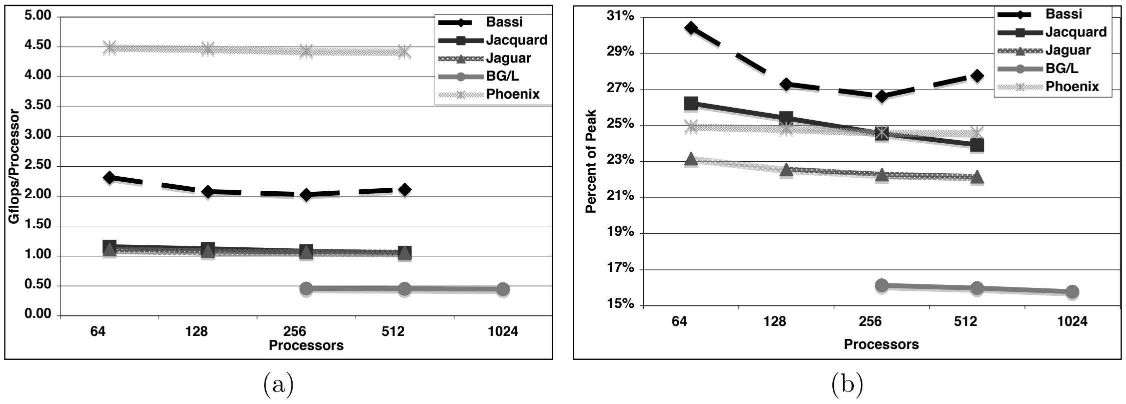


FIGURE 1.4: ELBM3D strong-scaling performance using a 512^3 grid by (a) GFlop/s and (b) percentage of peak. ALL BG/L data collected on the ANL BG/L system in coprocessor mode.

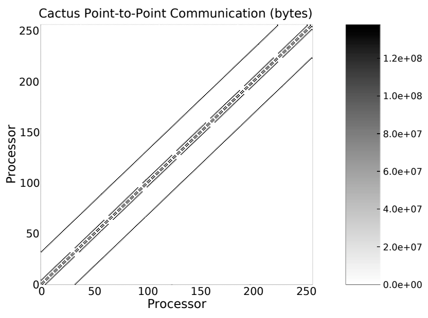
ACML for AMD — that compute values for a vector of arguments. (The benefits of these libraries are discussed in Section 1.4.) Using this approach gave ELBM3D a performance boost of between 15–30% depending on the architecture. For the X1E, the innermost gridpoint loop was taken inside the non-linear equation solver to enable full vectorization. After these optimizations, ELBM3D has a kernel of fairly high computational intensity and a percentage of peak of 15–30% on all architectures.

ELBM3D shows good scaling across all of our evaluated platforms. This is due to the dominance of nearest-neighbor point-to-point messaging, and lack of load balance issues. As expected, the parallel overhead increases as the ratio of communication to computation increases. The parallel efficiency as we move to higher concurrencies shows the least degradation on the BG/L system (although the memory requirements of the application and MPI implementation prevents running this size on fewer than 256 processors). Both Phoenix and Jaguar are very close behind, followed by Jacquard and Bassi.

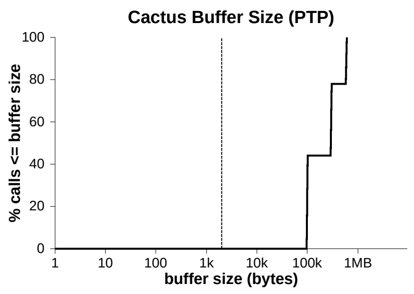
Our experiments bear out the fact that the higher computational cost of the entropic algorithm, as compared to traditional LBM approaches, can be cast in a way that leads to efficient computation on commodity processors. We are thus optimistic that ELBM3D will be able to deliver exceptional performance on planned petascale platforms.

1.6 Cactus: General Relativity Astrophysics

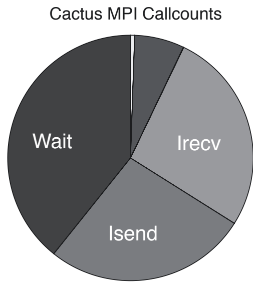
One of the most challenging problems in astrophysics is the numerical solution of Einstein's equations following from the Theory of General Relativity (GR): a set of coupled nonlinear hyperbolic and elliptic equations containing thousands of terms when fully expanded. The BSSN-MoL application, which makes use of the Cactus Computational ToolKit [3, 10], is designed to evolve Einstein's equations stably in 3D on supercomputers to simulate astrophysical phenomena with high gravitational fluxes – such as the collision of two black holes and the gravitational waves radiating from



(a)



(b)



(c)

FIGURE 1.5: CACTUS (a) communication topology and intensity (b) cumulative distribution function of point-to-point message buffer sizes and (c) breakdown of MPI calls counts.

that event. While Cactus is a modular framework supporting a wide variety of applications, this study focuses exclusively on the GR solver, which implements the ADM-BSSN (BSSN) formulation [1] with Method of Lines integration to enable stable evolution of black holes. For parallel computation, the grid is block domain decomposed so that each processor has a section of the global grid. The standard MPI driver (PUGH) for Cactus solves the PDE on a local grid section and then updates the values at the ghost zones by exchanging data on the faces of its topological neighbors in the domain decomposition.

In the topology chart of Figure 1.5(a), we see that the ghost-zone exchanges of Cactus result in communications with "neighboring" nodes, represented by diagonal bands. In fact, each node communicates with at most 6 neighbors due to the regular computational structure of the 3D stencil application. This topology is isomorphic to a mesh and should work well on mesh or toroidal interconnect topologies if mapped correctly. The communication is strongly bandwidth bound and dominated by point-to-point messaging as can be seen in Figures 1.5(b) and (c) respectively.

1.6.1 Experimental results

Figure 1.6 presents weak-scaling performance results for the Cactus BSSN-MoL application using an 60^3 per processor grid. In terms of raw performance, the Power5-based Bassi clearly outperforms any other systems, especially the BG/L where the Gflops/P rate and the percentage of peak performance is somewhat disappointing. The lower computational efficiency of BG/L is to be expected from the simpler dual-issue in-order PPC440 processor core. However, while the per-processor performance of BG/L is somewhat limited, the scaling behavior is impressive, achieving near perfect scalability for up to 16K processors. This is (by far) the largest Cactus scaling experiment to date, and shows extremely promising results. Due to memory constraints we could not conduct virtual node mode simulations for the 60^3 data set, however further testing with a smaller 50^3 grid shows no performance degradation for up to 32K (virtual node) processors. This strongly suggests that the Cactus application will scale to extremely high-concurrency, petascale systems. All of the experiments were performed with the default processor topology mapping where the communication topology matched the torus topology of the physical interconnect. Additional investigations with alternative processor topology mappings on the BG/L showed no significant effects.

The Opteron-based Jacquard cluster shows modest scaling, which is probably due to the (rela-

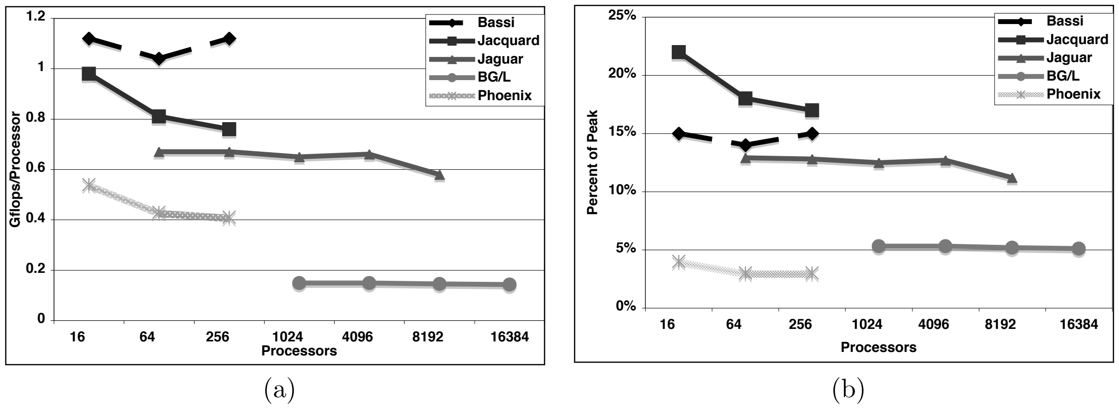


FIGURE 1.6: Cactus weak scaling experiments on a 60^3 per processor grid in (a) runtime and (b) percentage of peak. All BG/L data was run on the BGW system. Phoenix data collected on the Cray X1 platform.

tively) loosely coupled nature its system architecture as opposed to the tight software and hardware interconnect integration of the other platforms in our study. Separate studies of Jacquard InfiniBand interconnect also show increased messaging contention at higher concurrencies that are typical of multi-tier source-routed fat-tree networks [25]. Bassi shows excellent scaling but the size of the largest concurrency was significantly smaller compared to that of the BG/L so it remains to be seen if IBM's Federation HPS interconnect will scale to extremely large systems. Likewise, Jaguar, the Cray XT3 system, shows excellent scaling up to 8192 processors. The performance per processor for Jaguar's 2.6 GHz dual-core Opteron processors is somewhat lower than for Jacquard's single-core 2.2GHz Opteron system. The effect is primarily due to the differences in the quality of code generated by the compilers (PGI v1.5.31 on Jaguar and PathScale v2.4 on Jacquard). Moving from single core to dual core on Jaguar resulted in a modest 10% change in performance that was consistently less than the performance difference attributed to the compilers.

Phoenix, the Cray X1 platform, showed the lowest computational performance of our evaluated systems. The most costly procedure for the X1 was the computation of radiation boundary condition, which continued to drag performance down despite considerable effort to rewrite it in vectorizable form. In previous studies [5], the vectorized boundary conditions proved beneficial on a number of vector platforms including the NEC SX-8 and Earth Simulator; however the X1 continued to suffer disproportionally from small portions of unvectorized code due to the relatively large differential between vector and scalar performance, highlighting that notions of architectural balance cannot focus exclusively on bandwidth (bytes per flop) ratios.

1.7 PARATEC: First Principles Materials Science

PARATEC (PARAllel Total Energy Code [21]) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. The pseudopotentials are of

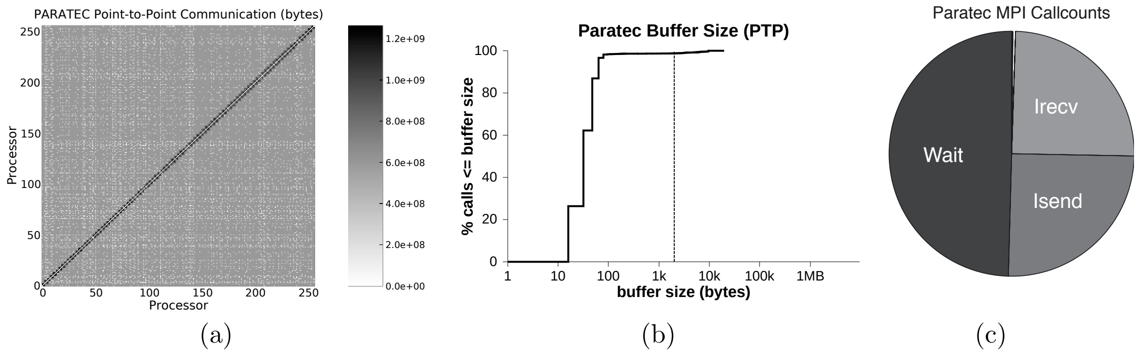


FIGURE 1.7: PARATEC (a) communication topology and intensity (b) cumulative distribution function of point-to-point message buffer sizes and (c) breakdown of MPI calls counts.

the standard norm-conserving variety. Forces can be easily calculated and used to relax the atoms into their equilibrium positions. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of Density Functional Theory (DFT) and obtain the ground-state electron wave functions. DFT is the most commonly used technique in materials science, incorporating a quantum mechanical treatment of the electrons to calculate the structural and electronic properties of materials. Codes based on DFT are widely used to study properties such as strength, cohesion, growth, magnetic, optical, and transport for materials like nanostructures, complex surfaces, and doped semiconductors.

PARATEC is written in F90 and MPI and is designed primarily for massively parallel computing platforms, but can also run on serial machines. The code has run on many computer architectures and uses preprocessing to include machine specific routines such as the FFT calls. Much of the computation time (typically 60%) involves FFTs and BLAS3 routines, which run at a high percentage of peak on most platforms. For small atomic systems, PARATEC tends to be dominated by the FFTs, and becomes dominated by the BLAS3 operations for larger atomic systems. The performance of the 3D FFTs tends to suffer at high concurrencies, but it can be controlled to a limited extent by using message aggregation (described below).

In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using parallel 3D FFTs to transform the wave functions between the two spaces. PARATEC uses its own handwritten 3D FFTs rather than library routines as the data layout in Fourier space is a sphere of points, rather than a standard square grid. The sphere is load balanced by distributing the different length columns from the sphere to different processors such that each processor holds a similar number of points in Fourier space. Effective load balancing is important, as much of the compute intensive part of the calculation is carried out in Fourier space [4].

Figure 1.7 shows the communication characteristics of PARATEC. The global data transposes within PARATEC's FFT operations — as captured in Figure 1.7(a) — account for the bulk of PARATEC's communication overhead, and can quickly become the bottleneck at high concurrencies. In general, with a fixed problem size the message sizes become smaller and increasingly latency bound at higher concurrencies, as can be seen in Figure 1.7(b). In a single 3D FFT the size of the data packets scale as the inverse of the number of processors squared. The PARATEC code can perform an all-band calculations that allow the FFT communications to be blocked together, resulting in larger message sizes and avoiding latency problems. However, the buffering required for such message aggregation consumes additional memory, which is a scarce resource on systems such as BG/L. Finally, observe in Figure 1.7(c) that the vast majority of messaging occurs in a point-to-point fashion.

1.7.1 Experimental results

Figure 1.8 presents strong-scaling performance results for a 488 atom CdSe (Cadmium Selenide) Quantum Dot (QD) system which has important technological applications due to its photoluminescent properties. Due to the use of BLAS3 and optimized one dimensional FFT libraries, which are highly cache resident, PARATEC obtains a high percentage of peak on the different platforms studied. The results for BG/L are for a smaller system (432 atom bulk silicon) due to memory constraints.

Results show that the Power5-based Bassi system obtains the highest absolute performance of 5.49 Gflops/P on 64 processors with good scaling to larger processor counts. The fastest Opteron systems (3.39 Gflops/P) was Jaguar (XT3) running on 128 processors. (Jacquard did not have enough memory to run the QD system on 128 processors.) The higher bandwidth for communications on Jaguar (see Table 1.1) allows it to scale better than Jacquard for this communication-intensive application. The BG/L system has a much lower single processor performance than the other evaluated platforms due to a relatively low peak speed of only 2.8 GF/s. BG/L's percent of peak drops significantly from 512 to 1024 processors, probably due to increased communication overhead when moving from a topologically packed half-plane of 512 processors to a larger configuration. The smaller system being run on the BG/L (432 atom bulk silicon) also limits the scaling to higher processor counts. Overall, Jaguar obtained the maximum aggregate performance of 4.02 Tflops on 2048 processors.

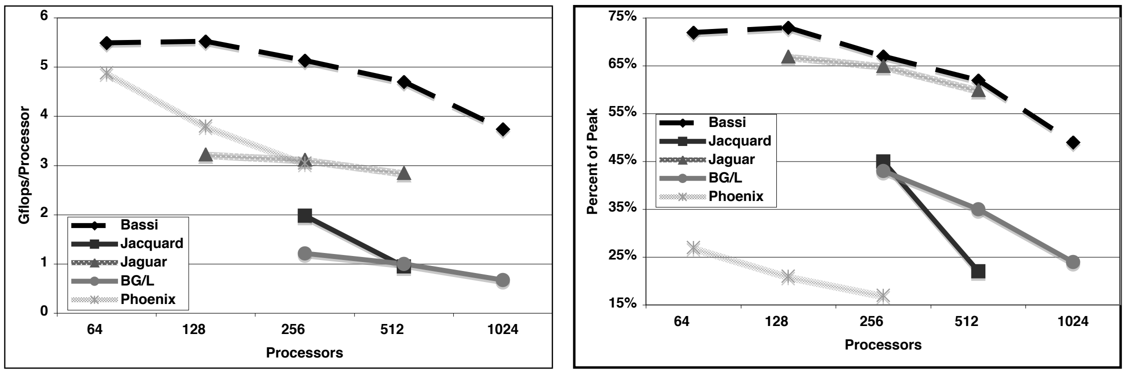


FIGURE 1.8: PARATEC strong-scaling performance on a 488 atom CdSe quantum dot. Power5 data for $P = 1024$ was run on LLNL's Purple system[†]. The BG/L data, collected on the BGW, is for a 432 atom bulk silicon due to memory constraints. Phoenix X1E data was collected with an X1 binary.

Looking at the vector system, results show that the Phoenix X1E achieved a lower percentage of peak than the other evaluated architectures; although in absolute terms, Phoenix performs rather well due to the high peak speed of the MSP processor*. One reason for this is the relatively slow performance of the X1E scalar unit compared to the vector processor. In consequence, the X1E spends a smaller percentage of the total time in highly optimized 3D FFTs and BLAS3 libraries than on any of the other machines. The other code segments are handwritten F90 routines and have a lower vector operation ratio.

PARATEC results do not show any clear advantage for a torus versus a fat-tree communication network. The main limit to scaling in PARATEC are the handwritten 3D FFTs, where all-to-all communications are performed to transpose the data across the machine. In a single 3D FFT the size of the data packets scales as the inverse of the number of processors squared. PARATEC can perform an all-band calculations, allowing the FFT communications to be blocked, resulting in larger message sizes and avoiding latency problems. Overall, the scaling of the FFTs is limited to a few thousand processors. Therefore, scaling PARATEC to petascale systems with tens (or hundreds) of thousands of processors requires a second level of parallelization over the electronic band indices as is done in the QBox code [11]. This will greatly benefit the scaling and reduce per processor memory requirements on architectures such as BG/L.

1.8 HyperCLaw: Hyperbolic AMR Gas Dynamics

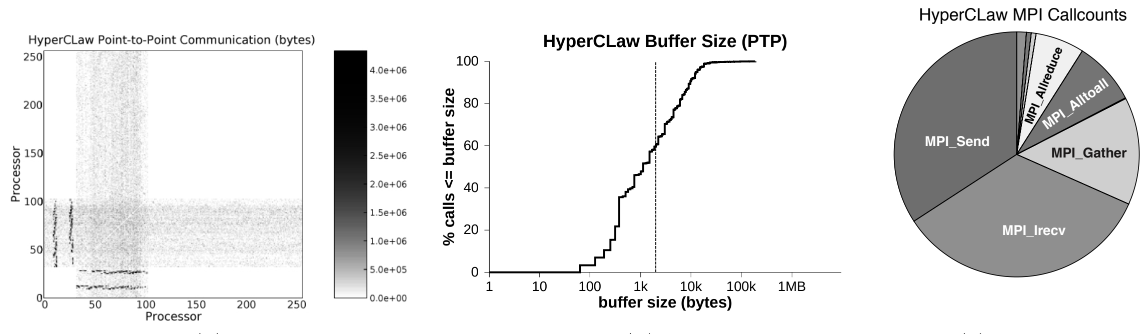
Adaptive mesh refinement (AMR) is a powerful technique that reduces the computational and memory resources required to solve otherwise intractable problems in computational science. The AMR strategy solves the system of partial differential equations (PDEs) on a relatively coarse grid, and dynamically refines it in regions of scientific interest or where the coarse grid error is too high for proper numerical resolution. HyperCLaw is a hybrid C++/`Fortran` AMR code developed and maintained by CCSE at LBNL [9, 22] where it is frequently used to solve systems of hyperbolic conservation laws using a higher-order Godunov method.

The HyperCLaw code consists of an applications layer containing the physics classes defined in terms of virtual functions. The basic idea is that data blocks are managed in C++ in which ghost cells are filled and temporary storage is dynamically allocated so that when the calls to the physics algorithms (usually finite difference methods implemented in `Fortran`) are made, the same stencil can be used for all points and no special treatments are required.

The HyperCLaw problem examined in this work profiles a hyperbolic shock-tube calculation, where we model the interaction of a Mach 1.25 shock in air hitting a spherical bubble of helium. This case is analogous to one of the experiments described by Haas and Sturtevant [12]. The difference between the density of the helium and the surrounding air causes the shock to accelerate into and then dramatically deform the bubble. The base computational grids for the problems

[†]Purple, located at LLNL, is architecturally similar to Bassi and contains 12,208 IBM Power5 processors. The authors thank Tom Spelce and Bronis de Supinksi of LLNL for conducting the Purple experiments.

*Results on the X1E were obtained by running the binary compiled on the X1, as running with an optimized X1E generated binary (-O3) caused the code to freeze. Cray engineers are investigating the problem.



(a) (b) (c)

FIGURE 1.9: HyperCLaw (a) communication topology and intensity (b) cumulative distribution function of point-to-point message buffer sizes and (c) breakdown of MPI calls counts.

studied is 512 × 64 × 32. These grids were adaptively refined by an initial factor of 2 and then a further factor of 4, leading to effective sizes of 2048 × 256 × 256 and 4096 × 512 × 256, respectively.

HyperCLaw is also dominated by point-to-point MPI messages with minimal collective communication requirements as shown in Figure 1.9(c). Most of the communication overhead occurs in the FillPatch operation. FillPatch starts with the computational grid at a given level, adds ghost cells five layers thick around each grid, and then fills those cells either by copying from other grids at that level, or by interpolating from lower level (coarser) cells that are covered by those ghost cells. FillPatch has a very complicated sparse nonlinear communication pattern as shown in Figure 1.9(a). Once it completes, all the ghost cells for each grid are filled with valid data, a higher-order Godunov solver is applied to each resulting grid.

Although the communication topology appears to be a many-to-many pattern that would require a high bisection bandwidth, the message sizes shown in Figure 1.9(b) are surprisingly small. Such small messages will have only ephemeral contention events and primarily be dominated by the interconnect latency and overhead of sending the message. Further analysis shows that if we focus exclusively on the messages that dominate the communication time, the number of *important* communicating partners drops significantly. Therefore, despite the complexity of the communication topology and large aggregate number of communicating partners, bisection bandwidth does not appear to be a critical factor in supporting efficient scaling of this AMR application. Architectural support for petascale AMR codes make therefore depend more on reducing the computational overhead of sending small messages. Understanding the evolving communication requirements of AMR simulations will be the focus of future work.

1.8.1 Experimental results

The HyperCLaw problem examined in this work profiles a hyperbolic shock-tube calculation, where we model the interaction of a Mach 1.25 shock in air hitting a spherical bubble of helium. This case is analogous to one of the experiments described by Haas and Sturtevant [12]. The difference between the density of the helium and the surrounding air causes the shock to accelerate into and then dramatically deform the bubble. The base computational grids for the problems studied is 512 × 64 × 32. These grids were adaptively refined by an initial factor of 2 and then a further factor of 4, leading to an effective resolution of 4096 × 512 × 256.

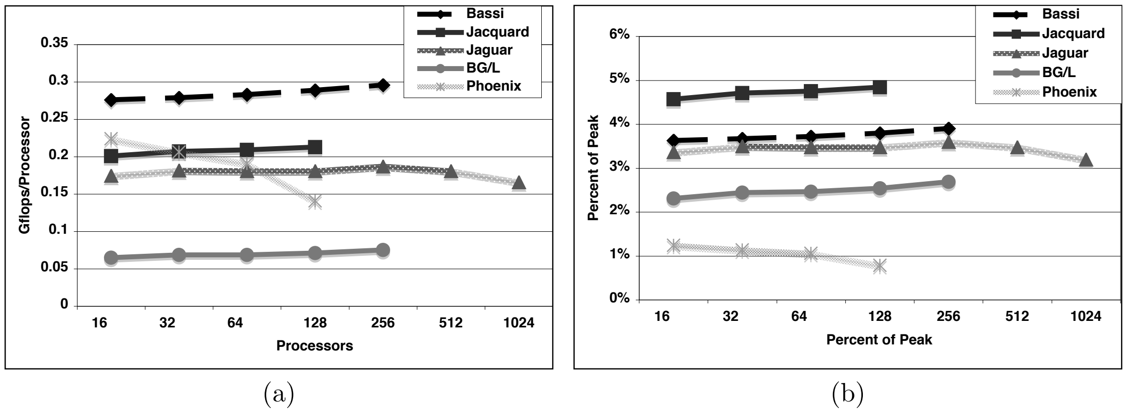


FIGURE 1.10: HyperCLaw weak-scaling performance on a base computational grid of 512x64x32 in (a) runtime and (b) percentage of peak. All BG/L data collected on ANL system (the BG/L system was unable to run jobs over 256 processors due to memory constraints).

Figure 1.10 presents the absolute runtime and percentage of peak for the weak-scaling HyperCLaw experiments. In terms of absolute runtime (at P=128), Bassi achieves the highest performance followed by Jacquard, Jaguar, Phoenix, and finally BG/L (the Phoenix and Jacquard experiments crash at P$\geq$256; system consultants are investigating the problems). Observe that all of the platforms achieve a low percentage of peak; for example at 128 processors, Jacquard, Bassi, Jaguar, BG/L, and Phoenix achieve 4.8%, 3.8%, 3.5%, 2.5%, and 0.8% respectively. Achieving peak performance on BG/L requires double-hummer mode (as described in Section 1.2), which has operand alignment restrictions that make it very difficult for the compiler to schedule efficiency for this application. Therefore BG/L delivered performance is most likely to be only half of the stated theoretical peak because of its inability to exploit the double-hummer. With this in mind, the BG/L would achieve a sustained performance of around 5%, commensurate with the other platforms in our study. Note that although these are weak-scaling experiments in the numbers of grids, the volume of work increases with higher concurrencies due to increased volume of computation along the communication boundaries; thus, the percentage of peak generally increases with processor count.

Although Phoenix performs relatively poorly for this application, especially in terms of its attained percentage of peak, it is important to point out that two effective X1E optimization were undertaken since our initial study into AMR vector performance [32]. Our preliminary study showed that the *knapsack* and *regridding* phases of HyperCLaw were largely to blame for limited X1E scalability, cumulatively consuming almost 60% of the runtime for large concurrency experiments. The original knapsack algorithm — responsible for allocating boxes of work equitably across the processors — suffered from a memory inefficiency. The updated version copies pointers to box lists during the swapping phase (instead of copying the lists themselves), and results in knapsack performance on Phoenix that is almost cost-free, even on hundreds of thousands of boxes.

The function of the regrid algorithm is to replace an existing grid hierarchy with a new hierarchy in order to maintain numerical accuracy, as important solution features develop and move through the computational domain. This process includes tagging coarse cells for refinement and buffering them to ensure that neighboring cells are also refined. The regridding phase requires the computations of box list intersection, which was originally implemented in a $\mathcal{O}(N^2)$ straightforward fashion. The updated version utilizes a hashing scheme based on the position in space of the bottom corners of the boxes, resulting in a vastly-improved $\mathcal{O}(NlogN)$ algorithm. This significantly reduced the cost of the regrid algorithm on Phoenix, resulting in improved performance. Nonetheless, Phoenix performance still remains low due to the non-vectorizable and short-vector-length operations necessary to maintain and regrid the hierarchical data structures.

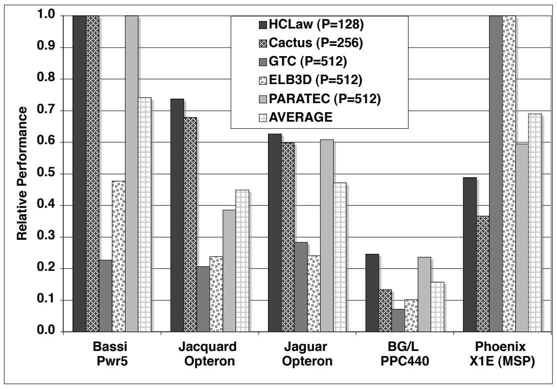
Overall, our HyperCLaw results highlight the relatively low execution efficiency of the AMR approach as measured by FLOP rate. This is due (in part) to the irregular nature of the AMR components necessary to maintain and regrid the hierarchical meshes, combined with complex communication requirements. Additionally, the numerical Godunov solver, although computationally intensive, requires substantial data movement that can degrade cache reuse. Nevertheless, the algorithmic efficiency gains associated with AMR and high-resolution discretizations more than compensate for the low sustained rate of execution as measured by FLOP rates. It points to the danger of measuring efficiency of an algorithm in terms of FLOP throughput alone rather than comparing performance the basis of time to solution. Other key result of our study are the knapsack and regridding optimizations, which significantly improved HyperCLaw scalability [32]. These improvements in scaling behavior suggest that, in spite of the low execution efficiency, the AMR methodology is a suitable candidate for petascale systems.

1.9 Summary and Conclusions

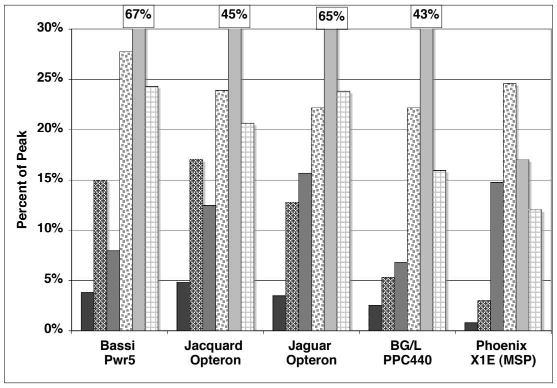
The purpose of any HEC system is to run full-scale scientific codes, and performance on such applications is the final arbiter of a platform's utility; comparative performance evaluation data must therefore be readily available to the HEC community at large. However, evaluating large-scale scientific applications using realistic problem sizes on leading supercomputing platforms is an extremely complex process, requiring coordination of application scientists in highly disparate areas. Our work presents one of the most extensive comparative performance results on modern supercomputers available in the literature.

Figure 1.11 shows a summary of results using the largest comparable concurrencies for all five studied applications and five state-of-the-art parallel platforms, in relative performance (normalized to the fastest system) and percentage of peak. Results show that the Power5-based Bassi system achieves the highest raw performance for three of our five applications, thanks to dramatically improved memory bandwidth (compared to its predecessors), and increased attention to latency hiding through advanced prefetch features. The Phoenix system achieved impressive raw performance on GTC and ELBM3D, however, application with nonvectorizable portions suffer greatly on this architecture due the imbalance between the scalar and vector processors. Comparing the two Opteron systems, Jacquard and Jaguar, we see that, in general, sustained performance is similar between the two platforms. However, for some applications such as GTC and PARATEC, the tight integration of Jaguar's XT3 interconnect results in significantly better scalability at high concurrency compared with Jacquard's commodity-based InfiniBand network. The BG/L platform attained the lowest raw and sustained performance on our suite of applications, however, results at very high concurrencies show impressive scalability characteristics and potential for attaining petascale performance.

Results also indicate that our evaluated codes have the potential to effectively utilize petascale



(a)



(b)

FIGURE 1.11: Summary of results for largest comparable concurrencies (a) relative runtime performance normalized to fastest system and (b) sustained percentage of peak. Cactus Phoenix results are on the X1 system. BG/L results are shown for P=1024 on Cactus and GTC, as smaller BG/L concurrencies are not available.

resources. However, some applications, such as PARATEC, will require significant reengineering to incorporate the additional levels of parallelism necessary to utilize vast numbers of processors. Others applications, including the lattice-bolzmann ELBM3D and the dynamically adapting HyperCLaw simulation, are already showing scaling behavior with promising prospects to achieve ultra-scale. Finally, two of our tested codes, Cactus and GTC, have successfully demonstrated impressive scalability up to 32K processors on the BGW system. A full GTC production simulation was also performed on 32,768 processors and showed a perfect load balance from beginning to end. This, combined with its high efficiency on multi-core processor, clearly puts GTC as a primary candidate to effectively utilize petascale resources.

Overall, these extensive performance evaluations are an important step toward conducting simulations at the petascale level, by providing computational scientists and system designers with critical information on how well numerical methods perform across state-of-the-art parallel systems. Future work will explore a wider set of computational methods, with a focus on irregular and unstructured algorithms, while investigating a broader set of HEC platforms, including the latest generation of multi-core technologies.

Acknowledgments

The authors would like to gratefully thank Bob Walkup for optimizing GTC on the BG/L as well as Tom Spelce and Bronis de Supinksi of LLNL for conducting the Purple experiments. We also thank Tom Goodale for his assistance with the Cactus code, Costin Iancu for his many thoughtful suggestions, and Arin Fishkin for her graphic contribution. The authors are grateful to IBM Watson Research Center for allowing BG/L access via the BGW Consortium Day. All LBNL authors were supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231. Dr. Ethier was supported by the Department of Energy under contract number DE-AC020-76-CH-03073.

References

[1] M. Alcubierre, G. Allen, B. Brügmann, et al. Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. *Phys. Rev. D*, (gr-qc/9908079), 2000.

[2] S. Ansumali and I. V. Karlin. Stabilization of the lattice boltzmann method by the h theorem: A numerical test. *Phys. Rev.*, E62:7999–8003, 2000.

[3] Cactus Code Server. `http://www.cactuscode.org`.

[4] A. Canning, L.W. Wang, A. Williamson, and A. Zunger. Parallel empirical pseudopotential electronic structure calculations for million atom systems. *J. Comp. Phys.*, 160, 2000.

[5] J. Carter, L. Oliker, and J. Shalf. Performance evaluation of scientific applications on modern parallel vector systems. In *VECPAR: High Performance Computing for Computational Science*, Rio de Janeiro, Brazil, July 10-12, 2006.

[6] T. H. Dunigan Jr., J. S. Vetter, J. B. White III, and P. H. Worley. Performance evaluation of the Cray X1 distributed shared-memory architecture. *IEEE Micro*, 25(1):30–40, Jan/Feb 2005.

[7] ORNL Cray X1 Evaluation. `http://www.csm.ornl.gov/~dunigan/cray`.

[8] S. Ethier, W.M. Tang, and Z. Lin. Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms. *J. Phys. : Conf. Series*, 16, 2005.

[9] Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory. `http://seesar.lbl.gov/CCSE`.

[10] T. Goodale, G. Allen, G. Lanfermann, et al. The Cactus framework and toolkit: Design and applications. In *VECPAR: 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer.

[11] F. Gygi, E. W. Draeger, Bronis R. de Supinski, et al. Large-scale first-principles molecular dynamics simulations on the BlueGene/L platform using the Qbox code. In *Proc. SC05: International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, Nov 12-18, 2005.

[12] J.-F. Haas and B. Sturtevant. Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities. *Journal of Fluid Mechanics*, 181:41–76, 1987.

[13] HPC challenge benchmark. `http://icl.cs.utk.edu/hpcc/index.html`.

[14] S. Kamil, L. Oliker, J. Shalf, and D. Skinner. Understanding ultra-scale application communication requirements. In *IEEE International Symposium on Workload Characterization*, 2005.

[15] Z. Lin, T. S. Hahm, W. W. Lee, et al. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, 281(5384):1835–7, Sep 1998.

[16] Z. Lin and W. W. Lee. Method for solving the gyrokinetic Poisson equation in general geometry. *Phys. Rev. E.*, 52:5646–5652, 1995.

[17] K. Nakajima. Three-level hybrid vs. flat mpi on the earth simulator: Parallel iterative solvers for finite-element method. In *Proc. 6th IMACS Symposium Iterative Methods in Scientific Computing*, volume 6, Denver, CO, Mar 27-30, 2003.

[18] L. Oliker, A. Canning, J. Carter, et al. Scientific application performance on candidate petascale platforms. In *Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Long Beach, CA, Mar 26-30, 2007.

[19] L. Oliker, A. Canning, J. Carter, J. Shalf, and S. Ethier. Scientific computations on modern parallel vector systems. In *Proc. SC04: International Conference for High Performance Computing, Networking, Storage and Analysis*, Pittsburgh, PA, Nov6-12, 2004.

[20] L. Oliker, J. Carter, M. Wehner, et al. Leading computational methods on scalar and vector HEC platforms. In *Proc. SC05: International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, Nov 12-18, 2005.

[21] PARAllel Total Energy Code. `http://www.nersc.gov/projects/paratec`.

[22] C. A. Rendleman, V. E. Beckner, M. L., W. Y. Crutchfield, et al. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3(3):147–157, 2000.

[23] SciDAC: Scientific Discovery through Advanced Computing. `http://www.scidac.gov/`.

[24] J. Shalf, S. Kamil, L. Oliker, and D. Skinner. Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect. In *Proc. SC05: International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, Nov 12-18, 2005.

[25] H. Shan, E. Strohmaier, J. Qiang, D. Bailey, and K. Yelick. Performance modeling and optimization of a high energy colliding beam simulation code. In *Proc. SC06: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2006.

[26] D. Skinner. Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, Heidelberg, Germany, 2005.

[27] STREAM: Sustainable memory bandwidth in high performance computers. `http://www.cs.virginia.edu/stream`.

[28] S. Succi. The lattice Boltzmann equation for fluids and beyond. *Oxford Science Publ.*, 2001.

[29] Top500 Supercomputer Sites. `http://www.top500.org`.

[30] G. Vahala, J. Yepez, L. Vahala, M. Soe, and J. Carter. 3D Entropic Lattice Boltzmann Simulations of 3D Navier-Stokes Turbulence. In *Proc. of 47th Annual Meeting of the APS Division of Plasma Physics*, Denver, CO, 2005.

[31] J. Vetter, S. Alam, T. Dunigan, Jr., et al. Early evaluation of the Cray XT3. In *Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, April 25-29, 2006.

[32] M. Welcome, C. Rendleman, L. Oliker, et al. Performance characteristics of an adaptive mesh refinement calculation on scalar and vector platforms. In *CF '06: Proceedings of the 3rd conference on Computing Frontiers*, May 2-5,2006.